

Robust Semi-Supervised Classification for Multi-Relational Graphs

Junting Ye Leman Akoglu
 Stony Brook University
 Department of Computer Science
 {juyye, leman}@cs.stonybrook.edu

Abstract

Graph-regularized semi-supervised learning has been used effectively for classification when (i) instances are connected through a graph, and (ii) labeled data is scarce. If available, using *multiple* relations (or graphs) between the instances can improve the prediction performance. On the other hand, when these relations have varying levels of veracity and exhibit varying relevance for the task, very noisy and/or irrelevant relations may deteriorate the performance. As a result, an effective weighing scheme needs to be put in place.

In this work, we propose a robust and scalable approach for multi-relational graph-regularized semi-supervised classification. Under a *convex optimization scheme*, we simultaneously infer weights for the multiple graphs as well as a solution. We provide a careful analysis of the inferred weights, based on which we devise an algorithm that filters out irrelevant and noisy graphs and produces *weights proportional to the informativeness* of the remaining graphs. Moreover, the proposed method is *linearly scalable* w.r.t. the number of edges in the union of the multiple graphs. Through extensive experiments we show that our method yields superior results under different noise models, and under increasing number of noisy graphs and intensity of noise, as compared to a list of baselines and state-of-the-art approaches.

1 Introduction

Given (1) a graph with *multiple* different relations between its nodes, and (2) labels for a small set of nodes, how can we predict the labels of the unlabeled nodes in a *robust* fashion? Robustness is a key element especially when the data comes from sources with varying veracity, where some relations may be irrelevant for the prediction task or may be too noisy.

This abstraction admits various real-world applications. For example, in fraud detection one may try to classify individuals as fraudulent or not based on the phone-call, SMS, financial, etc. interactions between them. In biology, genes are classified as whether or not they perform a certain function through various similarity relations between them. An example is shown in Figure 1, where a multi-graph with five different relation types G_1 - G_5 is depicted.

Accomplishing the above task requires addressing two main problems: (1) identifying and filtering out irrelevant

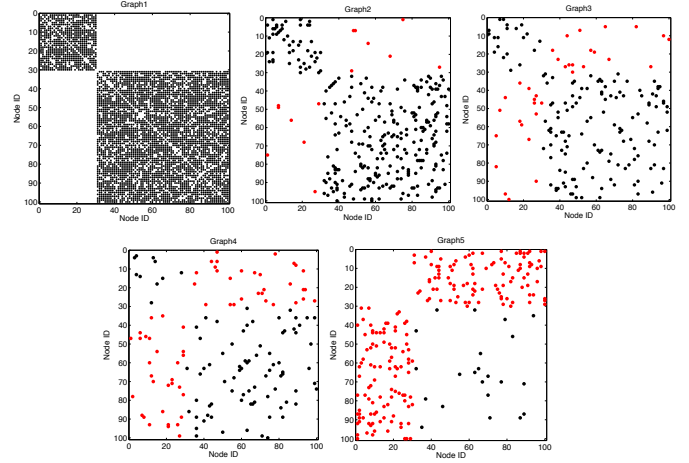


Figure 1: Example multi-graph ($n = 100$, $m = 5$), with 3 informative (top) and 2 intrusive (bottom) graphs. Shown are adjacency matrices, red dots depict cross-edges between nodes from different classes. G_1 - G_3 are in order of informativeness. G_4 depicts random noise. G_5 contains adversarial noise. Inferred weights (all graphs): $[25.17, 16.54, 12.79, 17.82, 27.68]$ (AP = 0.734 ± 0.035). Weights after noisy graphs removed: $[0.5000, 0.3003, 0.1997, 0, 0]$ (AP = 0.974 ± 0.004). AP: average precision.

and noisy relations, and (2) automatically weighing other relations by their informativeness for the task. Existing methods either are vastly affected in the presence of noise [17], produce locally optimal solutions due to their non-convex objective formulations [6, 13, 18], or are too expensive to compute [1, 7] (See Section 6).

In this work we introduce ROBUSTMULTISC, a robust, scalable, and effective semi-supervised classification approach for multi-relational graphs. In the example case, it recognizes G_4 and G_5 as irrelevant/noisy, and estimates optimal weights for relations G_1 - G_3 so as to leverage them collectively to achieve improved performance (See Figure 1 caption). Our contributions are listed as follows.

- **Model formulation:** Under a convex formulation, we simultaneously estimate weights for the multiple relations (or graphs) as well as a solution that utilizes a weighted combination of them.
- **Analysis of weights:** We show that in the presence of noise, the inferred weights reflect the impact of different

relations on the solution, where both dense informative and irrelevant/noisy graphs receive large weights.

- **Robustness:** Analysis of weights enable us to devise a robust algorithm that filters out irrelevant/noisy graphs, so as to produce weights proportional to the informativeness of relevant graphs.
- **Scalability:** Our proposed approach scales linearly w.r.t. the number of edges in the combined graph.
- **Effectiveness:** We show the efficacy of ROBUSTMULTISC on real-world datasets with varying number of relevant/noisy graphs, under different noise models, and varying intensity of noise, where it outperforms five baseline approaches including the state-of-the-art.

2 Problem Definition

In this work we consider real-world problem settings in which (1) the problem is cast as a binary classification task, (2) data objects are related through multiple different relationships, and (3) ground-truth class labels are scarce.

Motivating examples of this setting can be found particularly in anomaly detection (fraud, spam, etc.) and biological applications. For example, in bank fraud the goal is to classify users, i.e., account holders, as fraudulent or benign. These users may relate to one another through various relations, e.g., phone calls, emails, social links, shared loans, etc. Moreover, known labels especially for the positive class is scarce since fraud is rare. Besides, labeling a user as a fraudster or benign is a tedious process for domain analysts.

Similarly in biology, the problem of predicting gene function can be cast as a binary classification problem, in which a gene exhibits a certain function or not [12]. The genes can be associated in multiple ways through heterogeneous sources of genomic and proteomic data. For example, function tends to be shared among genes with similar patterns of expression, similar three-dimensional structures, similar chemical sensitivity, with products that interact physically, etc. [8]. Moreover, the labeled data is small, as the fraction of genes known to have a given function is relatively much smaller than the list of all genes. In general, obtaining ground-truth labels in biological applications is often challenging due to the substantial effort required to verify annotations through physical lab experiments.

Overall, the data can be represented as a *multi-graph*, in which the nodes represent the data objects (e.g., account holders or genes) and multiple sets of undirected edges between these nodes capture associations implied by the particular relations. We give a formal definition as follows.

DEFINITION 1. (MULTI-GRAPH) A *multi-relational graph* (or a *multi-graph*) $\mathcal{G}(V, \mathcal{E})$ consists of a set of graphs $\{G_1(V, E_1), G_2(V, E_2), \dots, G_m(V, E_m)\}$, on the same node set V , $|V| = n$. Undirected (weighted) edges $\mathcal{E} = \{E_1, \dots, E_m\}$ correspond to links implied by m different relation types, where we denote $|\mathcal{G}| = m$.

Using the various relationships between the data objects may provide more information for a given classification task, especially in the face of sparse input labels. Collectively, more accurate predictions can be made by combining these *multiple association networks*. On the other hand, it is not realistic to assume that all available relationships (i.e., graphs) between the data objects would be relevant for each particular prediction task. Relations that are either (i) too noisy, or (ii) irrelevant to a task should be significantly down-weighted. In this paper, we simply refer to all such graphs as *intrusive*. Filtering intrusive relations is especially important when the data sources cannot be carefully controlled—for example, when data is collected from various Web repositories with varying veracity, when graphs are constructed using various (not carefully chosen) similarity functions between the objects, etc. In addition, the relevant graphs may have varying degree of informativeness for a task, which necessitates a careful weighing scheme.

Overall, it is essential to build robust classification models that can effectively leverage multiple relationships by carefully weighing relevant graphs while filtering out the intrusive ones. Our work addresses this problem of Robust Semi-supervised Classification for Multi-Graphs (RSCM): Given a binary classification task, a multi-graph, and a (small) set of labeled objects, the goal is to build an effective classifier that is *robust to noisy and irrelevant data*. We give the formal problem definition as follows.

DEFINITION 2. (RSCM PROBLEM) Given a multi-graph $\mathcal{G}(V, \mathcal{E})$, $|\mathcal{G}| = m$, and a subset of labeled seed nodes $L \subset V$; devise a learning procedure to infer the labels of unlabeled nodes $V \setminus L$, which assigns an optimal set of weights $\mathbf{w} = \{w_1, \dots, w_m\}$ to individual graphs where (i) intrusive graphs are filtered (i.e., $w_k = 0$), and (ii) relevant graphs receive weights relative to their informativeness.

3 Robust SsC for Multi-Graphs: Formulation

In this section we describe our formulation for the RSCM PROBLEM and later provide a learning procedure in Section 4. Our formulation particularly focuses on the inference and interpretation of individual graph weights $\mathbf{w} = \{w_1, \dots, w_m\}$ for the different relations. Following on Definition 2 we first provide notation used in our formulation.

Notation. Let $L = \{v_1, \dots, v_l\}$ denote the set of labeled nodes, and $U = \{v_{l+1}, \dots, v_n\}$ denote the set of unlabeled nodes. We define a vector $\mathbf{y} = (y_1, \dots, y_l, y_{l+1}, \dots, y_n)$, where $y_i \in \{+1, -1\}$ for $1 \leq i \leq l$, indicating node belongs to positive or negative class, and $y_j = 0$ for $l+1 \leq j \leq n$ for unlabeled nodes.

The estimation of y_i from our model is denoted by f_i , $i = \{1, \dots, n\}$. Then, $\mathbf{f}_L = (f_1, \dots, f_l)$ denotes estimations of \mathbf{y}_L , and $\mathbf{f}_U = (f_{l+1}, \dots, f_n)$ denotes estimations of \mathbf{y}_U ; where $\mathbf{f} = (\mathbf{f}_L, \mathbf{f}_U)$.

Given a weighted graph $G_k(V, E_k)$, we denote its adjacency matrix by \mathbf{W}_k and its Laplacian matrix by $\mathbf{L}_k = \mathbf{D}_k^{-\frac{1}{2}}(\mathbf{D}_k - \mathbf{W}_k)\mathbf{D}_k^{-\frac{1}{2}}$, where \mathbf{D}_k is a diagonal matrix, in which $\mathbf{D}_k(i, i) = \sum_j \mathbf{W}_k(i, j)$ and 0 elsewhere.

Throughout text, we use lowercase bold letters for vectors, uppercase bold letters to denote matrices, and plain font for scalars. We use apostrophe (e.g., \mathbf{f}') to denote transpose.

3.1 Graph-Regularized Semi-supervised Classification Generalizing traditional semi-supervised learning objectives [23] to multi-graphs, we can write

$$(3.1) \quad \min_{\mathbf{f}} \|\mathbf{f} - \mathbf{y}\|_2^2 + \lambda \sum_{k=1}^m \mathbf{f}' \mathbf{L}_k \mathbf{f}$$

where λ is a regularization parameter. The first term enforces both fit to the training data and small norm on the estimation of unlabeled nodes, as we can write

$$\|\mathbf{f} - \mathbf{y}\|_2^2 = (\mathbf{f} - \mathbf{y})'(\mathbf{f} - \mathbf{y}) = \sum_{i=1}^l (f_i - y_i)^2 + \sum_{i=l+1}^n f_i^2,$$

since $y_i = 0$ for unlabeled nodes. The second term enforces smoothness with respect to the composite graph structure, since

$$\sum_{k=1}^m \mathbf{f}' \mathbf{L}_k \mathbf{f} = \sum_{k=1}^m \sum_{i,j \in V} \mathbf{W}_k(i, j) \left(\frac{f_i}{\sqrt{\mathbf{D}_k(i, i)}} - \frac{f_j}{\sqrt{\mathbf{D}_k(j, j)}} \right)^2.$$

Taking the derivative w.r.t. \mathbf{f} and setting it to zero, we obtain the solution

$$\mathbf{f}^* = (\mathbf{I} + \lambda \sum_k \mathbf{L}_k)^{-1} \mathbf{y}.$$

In the above derivation, all the graphs have equal impact on solution \mathbf{f} . To enable different impact (based on informativeness), one can instead use a *weighted* sum of the Laplacians in (3.1) and obtain

$$\mathbf{f}^* = (\mathbf{I} + \lambda \sum_k w_k \mathbf{L}_k)^{-1} \mathbf{y}.$$

Unfortunately, one often does not know the weights w_k a priori, which is also hard to set manually, especially for large multi-graphs with many relations. As such, one may try to infer both \mathbf{f} and \mathbf{w} by solving e.g., the following optimization problem:

$$(3.2) \quad \min_{\mathbf{f}, \mathbf{w}} \|\mathbf{f} - \mathbf{y}\|_2^2 + \lambda \sum_k \mathbf{f}' w_k \mathbf{L}_k \mathbf{f} \\ s.t. \quad w_k \geq 0, \sum_k w_k = 1$$

The above objective, however, is non-convex in both \mathbf{f} and \mathbf{w} . To get around this issue, several previous approaches have proposed alternating optimization schemes for similar objective functions [6, 18].

In this work, inspired by the TSS approach [17], we introduce a scheme that infers \mathbf{f} and \mathbf{w} *together* under a *convex* setup. The graph weights we infer (i.e., w_k 's) capture the *impact* that each graph has on the solution (i.e., on \mathbf{f}). Building on this interpretation, we devise a learning procedure that estimates \mathbf{f} which is *robust* to intrusive graphs. In the experiments, we show the superiority of our method to TSS [17], its robust extension [6], and others such as using equal weights (See Eq. (3.1)) or weights estimated solely based on labeled data [12] (See Section 5).

3.2 Objective Formulation We start by reorganizing the objective in Eq. (3.1) as

$$(3.3) \quad \min_{\mathbf{f}} (\mathbf{f} - \mathbf{y})'(\mathbf{f} - \mathbf{y}) \\ s.t. \quad \mathbf{f}' \mathbf{L}_k \mathbf{f} \leq c, \forall k = 1, \dots, m$$

The above aims to find a solution, where error of smoothness on each graph is bounded by a positive constant c (as $\mathbf{f}' \mathbf{L}_k \mathbf{f}$ is non-negative). Considering some graphs are less informative than others, we can allow for some slack, and rewrite

$$(3.4) \quad \min_{\mathbf{f}, \xi} (\mathbf{f} - \mathbf{y})'(\mathbf{f} - \mathbf{y}) + c_0 \sum_{k=1}^m \xi_k \\ s.t. \quad \mathbf{f}' \mathbf{L}_k \mathbf{f} \leq c + \xi_k, \xi_k \geq 0, \forall k = 1, \dots, m$$

By introducing the Lagrange multipliers (w_k 's, β_k 's), we get:

$$(3.5) \quad \min_{\mathbf{f}, \xi} \max_{\mathbf{w}, \beta} (\mathbf{f} - \mathbf{y})'(\mathbf{f} - \mathbf{y}) + c_0 \sum_{k=1}^m \xi_k \\ + \sum_{k=1}^m w_k (\mathbf{f}' \mathbf{L}_k \mathbf{f} - c - \xi_k) - \sum_k \beta_k \xi_k \\ s.t. \quad w_k \geq 0, \beta_k \geq 0, \forall k = 1, \dots, m$$

Setting derivative with respect to ξ_k to zero we get:

$$c_0 - w_k - \beta_k = 0$$

As such, $c_0 \geq w_k \geq 0$ as $\beta_k \geq 0$. Substituting $c_0 - w_k = \beta_k$ to (3.5), we get:

$$(3.6) \quad \max_{\mathbf{w}} \min_{\mathbf{f}} (\mathbf{f} - \mathbf{y})'(\mathbf{f} - \mathbf{y}) + \sum_{k=1}^m w_k (\mathbf{f}' \mathbf{L}_k \mathbf{f} - c) \\ s.t. \quad c_0 \geq w_k \geq 0, \forall k = 1, \dots, m$$

Note that in the above we can swap max and min in the optimization (3.5) due to the min-max theorem [14]. Setting the derivative with respect to \mathbf{f} to zero we have:

$$(3.7) \quad \mathbf{f} - \mathbf{y} + \sum_k w_k \mathbf{L}_k \mathbf{f} = 0 \Rightarrow \mathbf{f} = (\mathbf{I} + \sum_k w_k \mathbf{L}_k)^{-1} \mathbf{y}$$

Using (3.7) we can rewrite (3.6) as

$$\begin{aligned}
\max_{\mathbf{w}} \min_{\mathbf{f}} \quad & \mathbf{y}\mathbf{y}' + \mathbf{f}'(\mathbf{f} - 2\mathbf{y}) + \mathbf{f}'\left(\sum_k w_k \mathbf{L}_k\right)\mathbf{f} - \sum_k w_k c \\
& = -2\mathbf{f}'\mathbf{y} + \mathbf{y}'\mathbf{y} + \mathbf{f}'(\mathbf{I} + \sum_k w_k \mathbf{L}_k)\mathbf{f} - c \sum_k w_k \\
& = -2\mathbf{f}'\mathbf{y} + \mathbf{y}'\mathbf{y} + \mathbf{f}'\mathbf{y} - c \sum_k w_k \\
& = \mathbf{y}'\mathbf{y} - \mathbf{f}'\mathbf{y} - c \sum_k w_k \equiv \\
\max_{\mathbf{w}} \quad & \mathbf{y}'\mathbf{y} - \mathbf{y}'(\mathbf{I} + \sum_k w_k \mathbf{L}_k)^{-1}\mathbf{y} - c \sum_k w_k
\end{aligned}$$

Therefore, the dual problem becomes:

$$\begin{aligned}
(3.8) \quad & \min_{\mathbf{w}} \quad \mathbf{y}'(\mathbf{I} + \sum_k w_k \mathbf{L}_k)^{-1}\mathbf{y} + c\|\mathbf{w}\|_1 \\
& \text{s.t. } c_0 \geq w_k \geq 0, \forall k = 1, \dots, m
\end{aligned}$$

The dual program (3.8) is convex and can be solved (e.g., using the projected gradient descent method) to infer the graph weights \mathbf{w} . One can then plug in those weights directly into Eq. (3.7) to estimate \mathbf{f} . However, this procedure as followed in [17], yields inferior results in the presence of irrelevant and noisy graphs. This has to do with what the inferred weights capture. We discuss the interpretation of w_k 's in Section 3.3, which motivates our devised algorithm ROBUSTMULTISC in Section 4. Before, we provide a modification of our formulation in the face of class bias (i.e., imbalanced class distribution).

Class bias. When there is bias in the class distribution (without loss of generality, let '+1' and '-1' respectively depict minority and majority classes), we assign large penalties to the misclassification of '+1' instances. Otherwise, the minority class is largely ignored when optimizing the loss function. This is often the case in anomaly detection where the anomalous class is relatively much smaller (e.g., fraudsters are much fewer than benign users).

In semi-supervised learning, only part of the nodes are labeled for training, and the rest are unlabeled (depicted with '0'). For each node type ('+1', '0', '-1'), we assign a different penalty coefficient, c_+ , c_u , c_- respectively. Let \mathbf{C} be a $n \times n$ diagonal matrix, called the *class penalty matrix*, where $\mathbf{C}(i, i) = c_+$ if $y_i = 1$, c_- if $y_i = -1$, and c_u if $y_i = 0$. As such, the criterion in (3.4) can be reformulated as:

$$\min_{\mathbf{f}, \xi} (\mathbf{f} - \mathbf{y})' \mathbf{C} (\mathbf{f} - \mathbf{y}) + c_0 \sum_{k=1}^m \xi_k$$

Following the same derivation as before, we obtain the dual problem

$$\begin{aligned}
(3.9) \quad & \min_{\mathbf{w}} \quad \mathbf{y}' \mathbf{C} (\mathbf{C} - \sum_k w_k \mathbf{L}_k)^{-1} \mathbf{C} \mathbf{y} + c\|\mathbf{w}\|_1 \\
& \text{s.t. } c_0 \geq w_k \geq 0, \forall k = 1, \dots, m
\end{aligned}$$

and the solution

$$(3.10) \quad \mathbf{f}^* = (\mathbf{C} + \sum_k w_k \mathbf{L}_k)^{-1} \mathbf{C} \mathbf{y} .$$

3.3 Graph Weights Interpreted Next we provide a detailed discussion on the interpretation of the inferred weights by (3.9). In a nutshell, we show that in the presence of intrusive graphs, the weights do *not* reflect the relative *informativeness* of individual graphs—but rather the relative *impact* of each graph on the solution.

Ideally, we want to infer a weight w_k for each graph G_k proportional to its informativeness for the task, where the weights for intrusive graphs are zero. For example, in Figure 1 we illustrate a toy multi-graph with five views. The ideal weights would look like $w_1 > w_2 > w_3 > w_4 = w_5 = 0$. As we show in the following, however, the estimated weights should be interpreted carefully when we have intrusive graphs in the data.

Graphs G_k with larger $\mathbf{f}'\mathbf{L}_k\mathbf{f}$ tend to get larger w_k :

We have the dual problem $d(\mathbf{w})$ in (3.8) when learning the weights. We know from basic calculus that

$$(3.11) \quad \frac{\partial}{\partial x} Y^{-1} = -Y^{-1} \left(\frac{\partial}{\partial x} Y \right) Y^{-1} .$$

Thus we derive the derivative of $d(\mathbf{w})$ w.r.t w_k as

$$(3.12) \quad \frac{\partial d(\mathbf{w})}{\partial w_k} = -\mathbf{y}'(\mathbf{I} + \sum_{i=1}^m w_i \mathbf{L}_i)^{-1} \mathbf{L}_k (\mathbf{I} + \sum_{i=1}^m w_i \mathbf{L}_i)^{-1} \mathbf{y} + c$$

Since $\mathbf{f} = (\mathbf{I} + \sum_{i=1}^m w_i \mathbf{L}_i)^{-1} \mathbf{y}$, we obtain

$$(3.13) \quad \frac{\partial d(\mathbf{w})}{\partial w_k} = -\mathbf{f}'\mathbf{L}_k\mathbf{f} + c$$

Based on (3.13), we make the following inferences:

- The value of $\mathbf{f}'\mathbf{L}_k\mathbf{f}$ (i.e., the smoothness penalty) determines the changing rate in each dimension w_k of the gradient descent, i.e.,

$$w_k^{t+1} \leftarrow w_k^t - \frac{\partial d(\mathbf{w})}{\partial w_k} = w_k^t + \mathbf{f}'\mathbf{L}_k\mathbf{f} - c .$$

- For graphs with larger $\mathbf{f}'\mathbf{L}_k\mathbf{f}$, the rate is larger. As a result, those graphs tend to receive larger weights.

Further intuition as to why graphs with large $\mathbf{f}'\mathbf{L}_k\mathbf{f}$ tend to have large weights is as follows.

SVM intuition: Graphs that are important for the solution (correct instances) as well as those that are irrelevant (erroneous instances) all become support vectors, and receive non-zero dual coefficients (i.e., weights).

Constraints intuition: Graphs that “obstruct” the road to the solution more, i.e., cause the constraints ($\mathbf{f}'\mathbf{L}_k\mathbf{f} \leq c$) to be violated the more, will get larger w_k .

Both dense informative and intrusive graphs G_k has large $\mathbf{f}'\mathbf{L}_k\mathbf{f}$ —and hence large w_k : In this part, we first show that dense informative graphs have large $\mathbf{f}'\mathbf{L}_k\mathbf{f}$. Consider a graph with no noisy edges (i.e., no edges between nodes from different classes) but with high edge density

among nodes that belong to the same class. For such a graph, $\mathbf{f}'\mathbf{L}_k\mathbf{f} = \sum_{i,j \in V} \mathbf{W}_k(i,j) \left(\frac{f_i}{\sqrt{\mathbf{D}_k(i,i)}} - \frac{f_j}{\sqrt{\mathbf{D}_k(j,j)}} \right)^2$ can be large due to the numerous non-zero (although likely small) quadratic terms in the sum.

Next, we argue that it is not only the dense informative graphs that have large $\mathbf{f}'\mathbf{L}_k\mathbf{f}$, but *also the intrusive graphs*. This is mainly due to the many cross-edges that irrelevant and noisy graphs have between the nodes from different classes, which yield large quadratic terms.

We demonstrate this through the inferred weights on our example multi-graph in Figure 1. Notice that while the highly informative G_1 and G_2 receive large weight, the noisy graphs G_4 and G_5 also obtain comparably large weights.

Finally, we show that the graphs with larger weights have higher impact on the final solution.

The larger the w_k , the higher the impact of G_k on \mathbf{f} :

When w_k 's are fixed in (3.5), the term $w_k(\mathbf{f}'\mathbf{L}_k\mathbf{f} - c - \xi_k)$ needs to be smaller for larger w_k as we are to minimize w.r.t. \mathbf{w} . As a result, the graphs that have large weights “pull” \mathbf{f} towards themselves, i.e., force the estimations f_i 's to be smooth over their structure. As such, larger weight graphs have larger impact on the solution. This is also evident from $\mathbf{f} = (\mathbf{I} + \sum_k w_k \mathbf{L}_k)^{-1} \mathbf{y}$ in (3.7)—the larger the w_k , the more the \mathbf{L}_k is integrated into and influences the solution \mathbf{f} .

These arguments show that in general the estimated weights are not indicators of graph quality or informativeness but of impact that each graph has on the solution. When intrusive graphs are present in the data, they receive large weights. As a result they inflict large impact on the solution, by “pull”ing the solution toward fitting to their structure. As intrusive graphs have structure that is *not* compliant with the class labels, we would obtain a poor solution (note the low initial performance in Figure 1).

On the other hand, the inferred weights would be as desired in the *absence* of intrusive graphs. Then, the dense informative graphs get large weights and are the sole claimers of high impact on \mathbf{f} (cf. Figure 1). This suggests one needs to “weed out” the intrusive graphs to obtain reliable estimates. We introduce our proposed algorithm for this goal next.

4 Robust SsC for Multi-Graphs: Algorithm

To briefly reiterate Section 3.3, we find that graphs with large weights highly influence the solution. Moreover, intrusive graphs (if any) are among those with large weights and should be carefully filtered out to obtain a reliable solution.

In this section, we propose a robust algorithm for semi-supervised classification in multi-graphs. The pseudo-code is given in Algorithm 1. We first describe the steps of the algorithm and then provide details on the parameters and computational complexity.

The goal is to successfully remove the intrusive graphs. The main idea is to explore the search space through simu-

Algorithm 1 ROBUSTMULTISC (proposed algorithm for robust semi-supervised classification for multi-graphs)

Input: Multi-graph $\mathcal{G} = \{G_1, \dots, G_m\}$, labeled nodes L , initial temperature t , class penalty matrix \mathbf{C}

Output: Label estimations \mathbf{f}

```

1: Construct  $\mathbf{y}$  as described in Notation (§3)
2:  $best\mathbf{f} \leftarrow \emptyset$ ,  $bestP = 0$ ,  $m = |\mathcal{G}|$ ,  $Q \leftarrow \mathcal{G}$ 
3: while  $Q$  is not empty do
4:    $\mathcal{GS} \leftarrow dequeue(Q)$ 
5:    $cvP = \text{Compute cross validation performance of } \mathcal{GS}$ 
6:   if  $rand(0, 1) \leq \exp(\frac{cvP - bestP}{tm - |\mathcal{GS}| + 1})$  then
7:      $\mathbf{w}_{\mathcal{GS}} \leftarrow \text{Solve (3.9) using } \mathcal{GS} \text{ and input } \mathbf{C}$ 
8:      $\mathbf{f}_{\mathcal{GS}} \leftarrow \text{Compute solution using (3.10) and } \mathbf{w}_{\mathcal{GS}}$ 
9:     Cluster the weights:  $(W_s, W_l) \leftarrow 2\text{-means}(\mathbf{w}_{\mathcal{GS}})$ 
10:    for each  $G_k \in \mathcal{GS}$  for which  $w_k \in W_l$  do
11:       $v \leftarrow hash(\mathcal{GS} \setminus G_k)$ 
12:      if  $v$  is null then  $Q \leftarrow Q \cup \mathcal{GS} \setminus G_k$ 
13:    end for
14:    if  $cvP > bestP$  then  $best\mathbf{f} \leftarrow \mathbf{f}_{\mathcal{GS}}$ ,  $bestP = cvP$ 
15:  end if
16: end while
17: return  $best\mathbf{f}$ 

```

lated annealing by carefully removing large-weighted graphs one at a time. We start with introducing a queue of graph-sets, which initially includes the set of all graphs (line 2). We process the graph-sets in the queue one by one until the queue becomes empty (line 3). For each graph-set \mathcal{GS} that we dequeue (line 4), we compute its cross-validation performance cvP on the labeled data (line 5). In our experiments, we use average-precision (AP; area under the precision-recall curve) as our performance metric. This metric is more meaningful than accuracy, especially in the face of class bias (one can achieve high accuracy by always predicting the majority class).

We record the best AP as $bestP$ during the course of our search (line 14). With probability $\exp(\frac{cvP - bestP}{tm - |\mathcal{GS}| + 1})$, we “process” the graph-set in hand (lines 7-13, which we will describe shortly), otherwise we discard it. In line 6, $t \leq 1$ is the temperature parameter of simulated annealing and $(m - |\mathcal{GS}|)$ denotes the number of removed graphs from the original set. If the graph-set \mathcal{GS} in hand yields a cvP that is larger than $bestP$, we always process the set further, since when $(cvP - bestP) \geq 0$, $\exp(\frac{cvP - bestP}{tm - |\mathcal{GS}| + 1}) \geq 1$. On the other hand, if \mathcal{GS} yields inferior performance, we still process it with some probability that is proportional to the size of the graph-set. That is, the probability of processing a set decreases as they have more graphs removed from the original set. The probability is also inversely proportional to the performance distance $(cvP - bestP)$. The larger the gap, the higher the chance that \mathcal{GS} will be discarded.

Next we describe the steps to “process” a graph-set \mathcal{GS} .

We first solve the optimization problem (3.9) using \mathcal{GS} for the graph weights $\mathbf{w}_{\mathcal{GS}}$ and compute the solution based on $\mathbf{w}_{\mathcal{GS}}$ (lines 7-8). Next we cluster the weights into two groups, those with small weights W_s and those with large weights W_l (line 9). We know, through the analysis in §3.3, that intrusive graphs are *among* the large-weighted graphs. The issue is we do not know in advance which ones, as dense informative ones are likely to also belong to this group. As such, we create from \mathcal{GS} candidate graph-sets that contain all but each large-weighted graph and add those to the queue. Note that we maintain a hash table of the candidate graph-sets (line 11), so that we avoid re-considering the same sets that might be generated through different removal paths. At the end, we return the solution $bestf$ with the $bestP$.

4.1 Parameters Our algorithm expects two main parameters; the initial simulated annealing temperature t , and the class penalty matrix \mathbf{C} . We describe how we carefully set these in the following. Note that our objective function (3.9) involves two further parameters c and c_0 . Those are hyper-parameters, which we choose through cross-validation.

Initial temperature t . As we remove more and more graphs from the input multi-graph, the probability of further considering a set with inferior performance should decrease. That is when $d = (cvP - bestP) < 0$, $p = \exp(\frac{d}{t(m - |\mathcal{GS}| + 1)})$ should decrease as $r = (m - |\mathcal{GS}| + 1)$ increases. As such, we need $t \leq 1$.

Assume that we have an expected range $[m_l, m_u]$ for the number of intrusive graphs in the data where m_l and m_u respectively denote the minimum and maximum number. We would then want the probability $p = \exp(\frac{d}{tr})$ to approach zero as r gets closer to m_u even for a considerably small d . That is, as $r \rightarrow m_u$ and $0 > d \geq d_{thresh}$ for small d_{thresh} , we want $p_{thresh} > p > 0$ for small p_{thresh} .

Since $t = (\frac{d}{\ln p})^{\frac{1}{r}}$, the range for t satisfying the above constraints can be given as

$$(4.14) \quad t \in [(\frac{d_{thresh}}{\ln p_{thresh}})^{\frac{1}{m_u}}, (\frac{d_{thresh}}{\ln p_{thresh}})^{\frac{1}{m_l}}]$$

Empirically, we let $d_{thresh} = -0.1$ and $p_{thresh} = 0.01$. Thus if we expect $m_l = 5$ and $m_u = 10$, then the initial temperature is chosen randomly from $t \in [0.465, 0.682]$.

Class penalty matrix \mathbf{C} . As described in 3.2, we can normalize biased class distribution by assigning larger penalty to minority-class mis-classification. Recall that c_+ , c_u , c_- denote penalty coefficients for classes ‘+1’, ‘0’ (unlabeled), and ‘-1’, respectively. We set these parameters as $c_+ = 1 + const * \text{sign}(1 - 2f) * \max(f, 1 - f)$, $c_u = 1$, and $c_- = 1 + const * \text{sign}(2f - 1) * \max(f, 1 - f)$, where $const$ is a constant set to 0.7, and f is the fraction of class ‘+1’ instances in the labeled set. For example, if $f = 0.2$, then $c_+ = 1.56$, $c_2 = 1$, $c_- = 0.44$. One can also treat $const$ as a hyper-parameter and select it through cross-validation under a performance metric of interest.

4.2 Computational Analysis While our algorithm as presented uses a queue to process candidate graph-sets sequentially, it is amenable to parallelization. In fact, our publicly available source code employs a parallel implementation, see www.cs.stonybrook.edu/~juyye/#code.

In particular, we can think of the explored graph-sets to form a search tree structure, rooted at the original multi-graph containing all the graphs. Each removal of a graph from a given graph-set produces a new leaf node in the tree attached to its superset. As such, a series of successive removals forms a path from the root to a leaf. Each of these search paths can be executed independently in parallel ($bestP$ and the hash table are shared across processes).

As such, the computational complexity is proportional to the *depth* of the search tree. Since we control the temperature parameter t such that the exploration probability approaches zero as we remove an expected maximum number m_u of graphs, we have $depth \leq m_u$.

At each node of the tree, we solve the optimization problem (3.9) using projected gradient descent, where the main computation involves computing the gradient (See (3.12) in §3.3). The gradient involves the term $(\mathbf{I} + \sum_{i=1}^m w_i \mathbf{L}_i)^{-1}$, i.e., the inverse of a $(n \times n)$ matrix which is $O(n^3)$ if done naively. The same is true for the solution \mathbf{f} which requires a similar inverse operation (See (3.7) or (3.10)). Luckily, however, we do not need to compute the inverse explicitly, because it always appears as the vector $\mathbf{x} = (\mathbf{I} + \sum_{i=1}^m w_i \mathbf{L}_i)^{-1} \mathbf{y}$. We can obtain \mathbf{x} as a solution of sparse linear systems, where the computational cost of the derivative is nearly linear in the number of non-zero entries of $\sum_{i=1}^m w_i \mathbf{L}_i$, i.e., proportional to the number of edges of the multi-graph [15].

Computing the dual objective then takes $O(s|\mathcal{E}|)$, where s is the number of steps of the gradient descent algorithm. All in all, the total time complexity of a parallel implementation becomes $O(s|\mathcal{E}|m_u)$, which is tractable for most sparse real-world multi-graphs where m_u and s are often small.

Pruning computation. In the following, we describe several strategies we use that help us keep the size of the search tree tractable, even for a serial implementation.

- **Branching factor:** During our search, we only consider graphs with large weights as candidates to remove. This way the branch-out factor of the search tree becomes smaller than considering all possible removals.
- **Simulated annealing:** Our search terminates when there is significant decrease in the cross-validation performance. As such, some search paths take shorter than others, i.e., the search tree is not fully-balanced.
- **Tree depth:** As discussed earlier, we control the depth of the search tree through m_u , expected upper bound on the number of intrusive graphs. As often $m_u \ll m$, the search tree is terminated significantly earlier than e.g., a brute-force search.

5 Evaluation

We evaluated our ROBUSTMULTISC on real-world datasets, and compared it to a list of baselines and state-of-the-art methods. We also “noise-tested” all methods under varying level, intensity, and models of noise.

5.1 Experiment Setup

Datasets. The real-world multi-graphs used in our work are publicly available, as listed in Table 1. *RealityMining* [5] contains 4 different relations between two classes of students (in business school and in computer science): phone call, Bluetooth scans, SMS, and friendship. *Protein* [17] consists of Yeast proteins, associated through 5 relations, where those with function “transport facilitation” constitute the positive class, and others the negative. *Gene1* and *Gene2* contain different sets of Yeast genes, each associated through 15 different genomic sources. Those are obtained from [12] which we refer to for details. Also see Appendix A for further description and visualization of our multi-graphs.

Baselines. We compare ROBUSTMULTISC against three state-of-the-art: TSS [17], ROBUSTLP [6], and GENEMANIA [12]. We also introduce two simple baselines, EQL-WGHT that assigns equal weight to all graphs and PERF-WGHT that assigns weights proportional to the cross-validation accuracy of individual graphs on labeled nodes.

Noise-testing. To test the robustness of the methods, we tested them in the presence of injected intrusive graphs with varying level, model, and intensity of noise.

- *Number of intrusive graphs:* We tested the effect of increasing noise level by injecting 2, 4 and 6 intrusive graphs at a time.
- *Noisy graph models:* We adopted 3 strategies to generate intrusive graphs, (1) Erdos-Renyi random graphs (ER), (2) edge-rewired original graphs (RW), and what we call (3) adversarial graphs (AV) (where most edges are cross-edges between the different classes).
- *Noise intensity (low/high):* Intensity reflects injected graph density (5%/50%) for ER, ratio of within-class edges rewired to be cross-edges (60%/80%) for RW, and ratio of cross-edges (60%/80%) for the AV model.

Overall, there are 3 different number of injected graphs, 3 models, and 2 noise intensities. As such, we “noise-tested” the methods under 18 ($3 \times 3 \times 2$) different settings.

5.2 Evaluation Results To perform semi-supervised classification, we label 5% of the nodes in *Protein*, *Gene1*, and *Gene2* and 30% in RM as it is a smaller dataset. We randomly sample the labeled set 10 times, and report the mean Average Precision (area under precision-recall curve) in Table 2 (notice in Table 1 that datasets are class imbalanced, hence accuracy is not a good measure to report). Also see Appendix B for the precision-recall plots.

We see that ROBUSTMULTISC outperforms all baselines across all the datasets. Its superior performance is espe-

Table 1: *Real-world multi-relational graphs used in work.*

Dataset	#Graphs m	#Nodes n	#Pos.	#Neg.
<i>RealityMining</i> [5]	4	78	27	51
<i>Protein</i> [17]	5	3,588	306	3,282
<i>Gene1</i> [12]	15	1,724	185	1,539
<i>Gene2</i> [12]	15	3,146	214	2,932

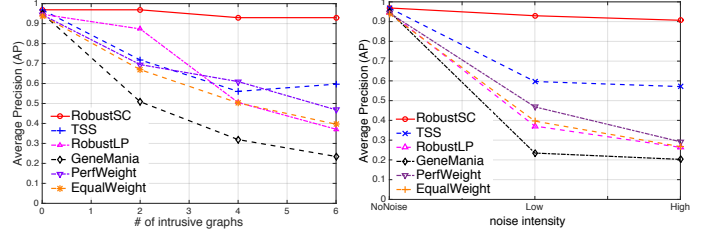


Figure 2: *Performance by (left) increasing number of intrusive graphs, and (right) increasing intensity of noise.*

cially evident in the presence of noise, when the performance of others degrade dramatically (See Appendix C for similar results on RM injected with 2 and 4 intrusive graphs).

We further investigate the affects of noise using *RealityMining* as a running example, as in the absence of noise all methods perform similarly on this multi-graph. Figure 2 (left) shows how the performance of the methods change with increasing number of intrusive graphs (under rewiring and low-intensity). Figure 2 (right) shows the same with different noise intensity (under rewiring, 6 intrusive graphs). These clearly show that the competing methods are hindered by noise, while ROBUSTMULTISC’s performance remains near-stable. In fact, as Figure 3 shows ROBUSTMULTISC is robust under all settings; increasing level and intensity as well as different models of noise.

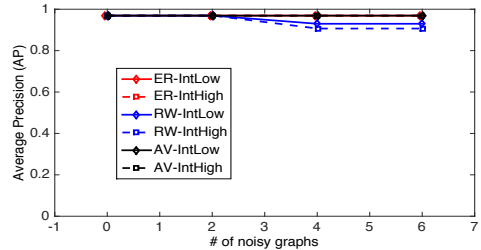


Figure 3: *ROBUSTMULTISC is robust under varying level (#graphs), intensity (low/high), models (ER,RW,AV) of noise.*

Figure 4 illustrates the removal order of graphs during the course of ROBUSTMULTISC (Algorithm 1), for each of the six noise settings for *RealityMining* in Table 2. The graph ID removed at each iteration is depicted in the figure. We see that simulated-annealing continues to remove graphs as long as cross-validation performance either increases or drops slightly, and stops when it drops significantly. Notice that the proposed method successfully removes most of the injected intrusive graphs G_5 - G_{10} . (See Appendix D for a similar figure for the real datasets).

Table 2: Performance of all methods on real-world multi-graphs. RM also injected with 6 intrusive graphs with various settings. Values depict mean and standard deviation Average Precision (AP) (over 10 runs with different labeled set).

Dataset	#Graphs	NoiseModel	Intensity	PROPOSED	PERF-WGHT	EQL-WGHT	TSS	ROBUSTLP	GENEMANIA
RM	4	—	—	0.970±0.004	0.944±0.004	0.939±0.018	0.970±0.004	0.947±0.010	0.951±0.021
	4+6	Adversarial	Low	0.970±0.004	0.389±0.049	0.277±0.024	0.354±0.036	0.284±0.026	0.217±0.012
	4+6	Adversarial	High	0.970±0.004	0.257±0.026	0.223±0.011	0.577±0.041	0.225±0.010	0.197±0.003
	4+6	Rewire	Low	0.930±0.004	0.468±0.049	0.396±0.033	0.597±0.041	0.371±0.036	0.235±0.011
	4+6	Rewire	High	0.907±0.073	0.292±0.026	0.267±0.020	0.571±0.044	0.264±0.023	0.202±0.004
	4+6	Erdos-Renyi	Low	0.970±0.004	0.860±0.029	0.756±0.050	0.494±0.072	0.810±0.043	0.645±0.077
	4+6	Erdos-Renyi	High	0.970±0.004	0.937±0.002	0.896±0.036	0.621±0.078	0.907±0.033	0.773±0.085
Protein	5	—	—	0.457±0.065	0.452±0.067	0.441±0.070	0.457±0.065	0.439±0.063	0.424±0.058
Gene1	15	—	—	0.703±0.056	0.658±0.048	0.632±0.053	0.648±0.059	0.628±0.054	0.509±0.077
Gene2	15	—	—	0.838±0.031	0.830±0.048	0.809±0.046	0.734±0.080	0.460±0.072	0.229±0.069

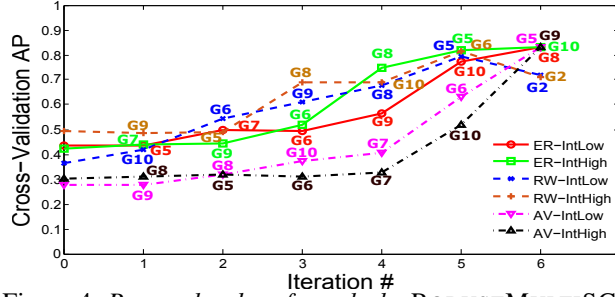


Figure 4: Removal order of graphs by ROBUSTMULTISC.

We also analyze the inferred weights by each method. Figure 5 shows the normalized weights on *RealityMining* with 6 injected graphs, under AV with high intensity. (Weight figures for all ten datasets in Table 2 are in Appendix E). Notice that all competing methods give non-zero weights to all the injected graphs G_5 - G_{10} , which hinders their performance. In contrast, ROBUSTMULTISC puts non-zero weight only on the informative graphs G_1 - G_4 , particularly large weights on the first two. These are in fact the well-structured and denser informative graphs (See Appendix A.1).

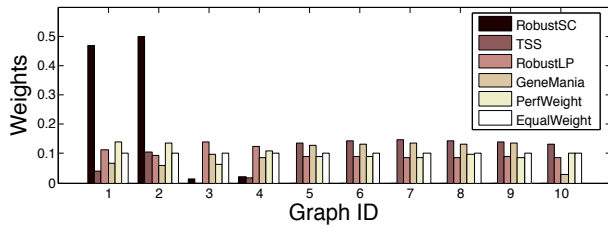


Figure 5: Inferred graph weights on RealityMining (+6 injected graphs G_5 - G_{10} under AV and high-intensity).

Finally in Figure 6 we show how the performance of the methods change when we increase the labeled set percentage in *RealityMining* from 30% up to 90% (6 injected graphs, under rewiring with low intensity; results are avg'd over 10 runs). As expected the performance improves for all methods with increasing labeled data. However, the competing methods cannot achieve improved robustness and reach the same performance level by ROBUSTMULTISC, even when they are provided 90% data labeled. As such, robustness is not a function of labeled data consumed.

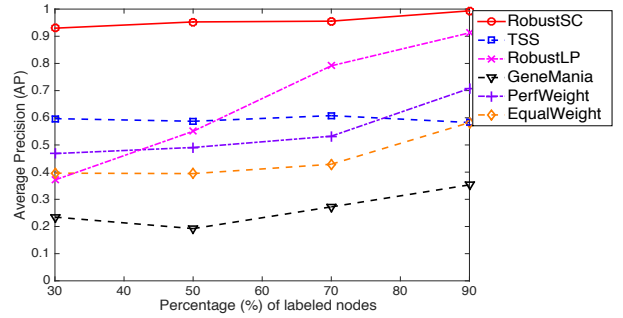


Figure 6: Performance vs. % labeled nodes. Competing methods remain hindered by noise despite 90% data labeled.

6 Related Work

Semi-supervised classification for graph data aims to classify the unlabeled nodes using a (small) set of available labeled nodes in the graph. Most methods in the literature (especially for unattributed graphs) are transductive, where the inference is performed over the graph without learning any particular classifier model [10, 4, 23, 21, 3]. These methods solve a global objective function that enforces smoothness of labels over the graph structure as well as goodness-of-fit to the labeled examples. For detailed review of graph-based semi-supervised learning, we refer to [22] and [16].

While most transductive methods consider a single given graph, there also exist methods that aim to predict the node labels by leveraging multiple graphs [7, 1, 17, 12, 6, 13]. The problem is sometimes referred as multiple kernel learning [2], where data is represented by means of kernel matrices defined by similarities between data pairs.

The SDP/SVM method by Lanckriet *et al.* [7] use a weighted sum of kernel matrices, and find the optimal weights within a semi-definite programming (SDP) framework. Argyriou *et al.* [1] build on their method to instead combine Laplacian kernel matrices. Multiple kernel learning has also been used for applications such as protein function prediction [20] and image classification and retrieval [19]. The SDP/SVM method however is cubic in data size and hence not scalable to large datasets. Moreover it requires valid (i.e., positive definite) kernel matrices as input.

Mostafavi *et al.* [12, 11] construct the optimal composite

graph by solving a linear regression problem using labeled data, later also used by Luo *et al.* [9]. Different from most semi-supervised transductive methods that leverage both labeled and unlabeled data, these use only the labeled data to estimate the graph weights.

The TSS method by Tsuda *et al.* [17] formulates label inference and weight estimation as a convex optimization problem. The weight assignment mechanism is close to the way that SVM selects support vectors. Kato *et al.* [6] show that the TSS algorithm produces inferior results when some graphs are irrelevant to the task and hence is not robust when noise is present. They propose a robust method, which alternates between minimizing their new (nonconvex) objective function with respect to label assignment and with respect to weight estimation. Similar alternating optimization of label and weight learning has also been used in [13] and [18], where slightly different objectives and weight update rules are used. All of these leverage both labeled and unlabeled data to estimate the graph weights. However, as they introduce nonconvex formulations, they are not guaranteed to find their globally optimum solution.

Different from the alternating optimization approaches in [6, 13, 18], we estimate the weights directly by solving a single optimization problem. Moreover, we iteratively discard the irrelevant graphs based on a simulated annealing approach guided by the cross-validation performance to prevent them from deteriorating the results. Through weeding out the intrusive graphs and learning globally optimal graph weights we produce the most robust results.

7 Conclusion

In this work we introduce ROBUSTMULTISC, for robust, scalable, and effective *semi-supervised transductive classification for multi-relational graphs*. The proposed method employs a *convex* formulation that estimates weights for individual graphs, along with a solution that utilizes a weighted combination of them. Based on the analysis of weights, we devise a new scheme that iteratively discards intrusive graphs to achieve *robust* performance. Moreover, ROBUSTMULTISC is *linearly scalable* w.r.t. the size of the combined graph. Extensive experiments on real-world multi-graphs show that ROBUSTMULTISC produces competitive results under varying level, intensity, and models of noise where it outperforms several baselines and state-of-the-art methods significantly, which are hindered by the presence of noise.

References

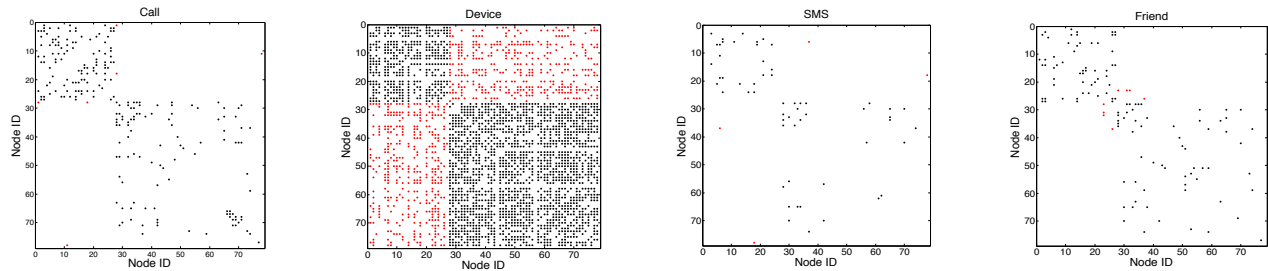
- [1] A. Argyriou, M. Herbster, and M. Pontil. Combining graph laplacians for semi-supervised learning. In *NIPS*, 2005.
- [2] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *ICML*, volume 69, 2004.
- [3] M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. In *COLT*, 2004.
- [4] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *ICML*, pages 19–26, 2001.
- [5] N. Eagle, A. S. Pentland, and D. Lazer. Inferring friendship network structure by using mobile phone data. *PNAS*, 106(36):15274–15278, 2009.
- [6] T. Kato, H. Kashima, and M. Sugiyama. Robust label propagation on multiple networks. *IEEE Transactions on Neural Networks*, 20(1):35–44, 2009.
- [7] G. R. G. Lanckriet, T. D. Bie, N. Cristianini, M. I. Jordan, and W. S. Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004.
- [8] G. R. G. Lanckriet, M. Deng, N. Cristianini, M. I. Jordan, and W. S. Noble. Kernel-based data fusion and its application to protein function prediction in yeast. In *Pacific Symposium on Biocomputing*, pages 300–311. World Scientific, 2004.
- [9] C. Luo, R. Guan, Z. Wang, and C. Lin. Hetpathmine: A novel transductive classification algorithm on heterogeneous information networks. In *Adv. in Info. Ret.* 2014.
- [10] S. Macskassy and F. Provost. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, 8:935–983, 2007.
- [11] S. Mostafavi and Q. Morris. Fast integration of heterogeneous data sources for predicting gene function with limited annotation. *Bioinformatics*, 26(14):1759–1765, 2010.
- [12] S. Mostafavi, D. Ray, D. Warde-Farley, C. Grouios, and Q. Morris. GeneMANIA: a real-time multiple association network integration algorithm for predicting gene function. *Genome Biology*, 9(Suppl 1):S4, 2008.
- [13] H. Shin, K. Tsuda, and B. Schölkopf. Protein functional class prediction with a combined graph. *Expert Syst. Appl.*, 36(2):3284–3292, 2009.
- [14] M. Sion. On general minimax theorems. *Pac. J. of Math.*, 8(1):171–176, 1958.
- [15] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC*, pages 81–90. ACM, 2004.
- [16] A. Subramanya and P. P. Talukdar. *Graph-Based Semi-Supervised Learning*. Synthesis Lect. on AI and ML. Morgan and Claypool Publishers, 2014.
- [17] K. Tsuda, H. Shin, and B. Schölkopf. Fast protein classification with multiple networks. *Bioinformatics*, 21:59–65, 2005.
- [18] M. Wan, Y. Ouyang, L. Kaplan, and J. Han. Graph regularized meta-path based transductive regression in heterogeneous information network. In *SDM*. SIAM, 2015.
- [19] S. Wang, S. Jiang, Q. Huang, and Q. Tian. S3MKL: scalable semi-supervised multiple kernel learning for image data mining. In *ACM Multimedia*, pages 163–172. ACM, 2010.
- [20] G.-X. Yu, H. Rangwala, C. Domeniconi, G. Zhang, and Z. Zhang. Protein function prediction by integrating multiple kernels. In *IJCAI*, 2013.
- [21] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, pages 321–328, 2003.
- [22] X. Zhu. Semi-supervised learning literature survey. Technical report, CS, U of Wisconsin-Madison, 2008.
- [23] X. Zhu, Z. Ghahramani, J. Lafferty, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 3, pages 912–919, 2003.

Appendix

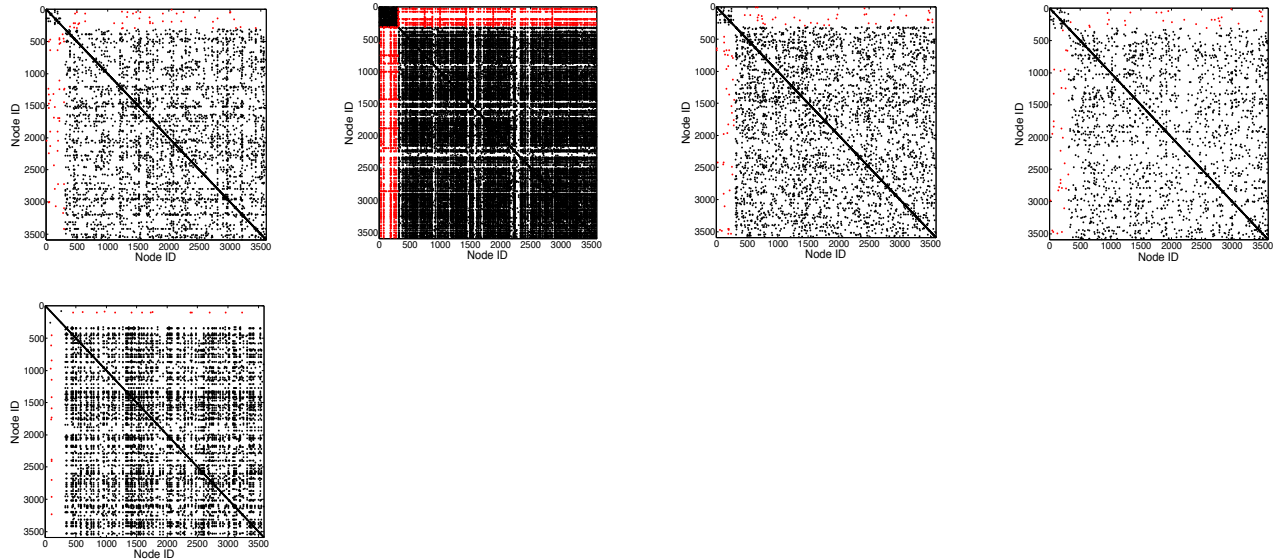
A Description and Visualization of Multi-Graphs

We visualize the adjacency matrices of individual graphs in our real-world multi-graphs as follows. The rows are ordered by class label, such that positive-class instances are listed first, followed by negative-class instances. Edges between the nodes are shown with dots, where black dots depict within-class edges and red dots depict cross-edges.

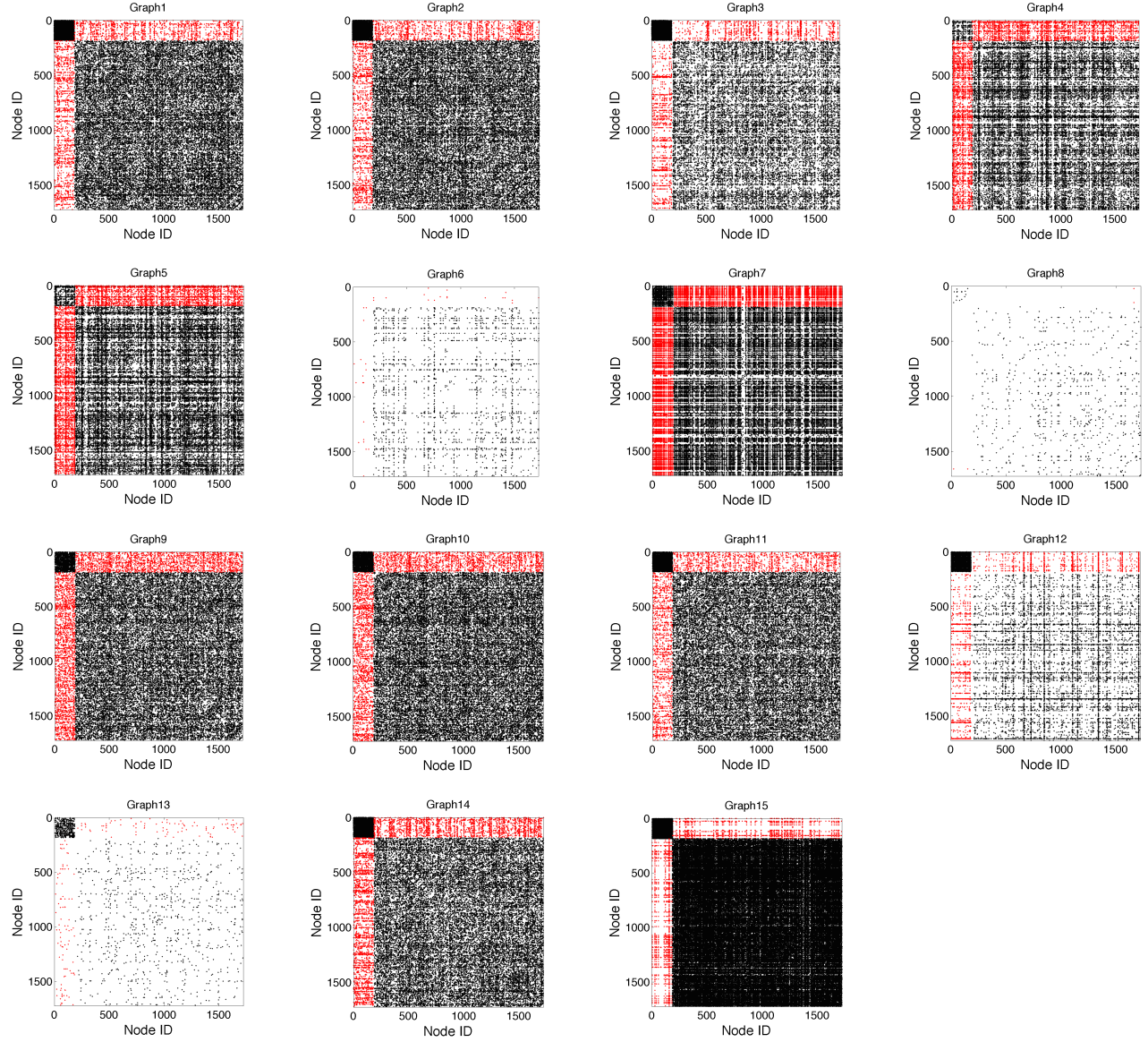
A.1 RealityMining *RealityMining* [5] is a 4-view dataset collected through tracking activities in students' cellphones. Two classes of students (business school students as positives and CS students as negatives) are linked via 4 types of activities: phone calls, Bluetooth device scans, text message exchanges and friendship claims.



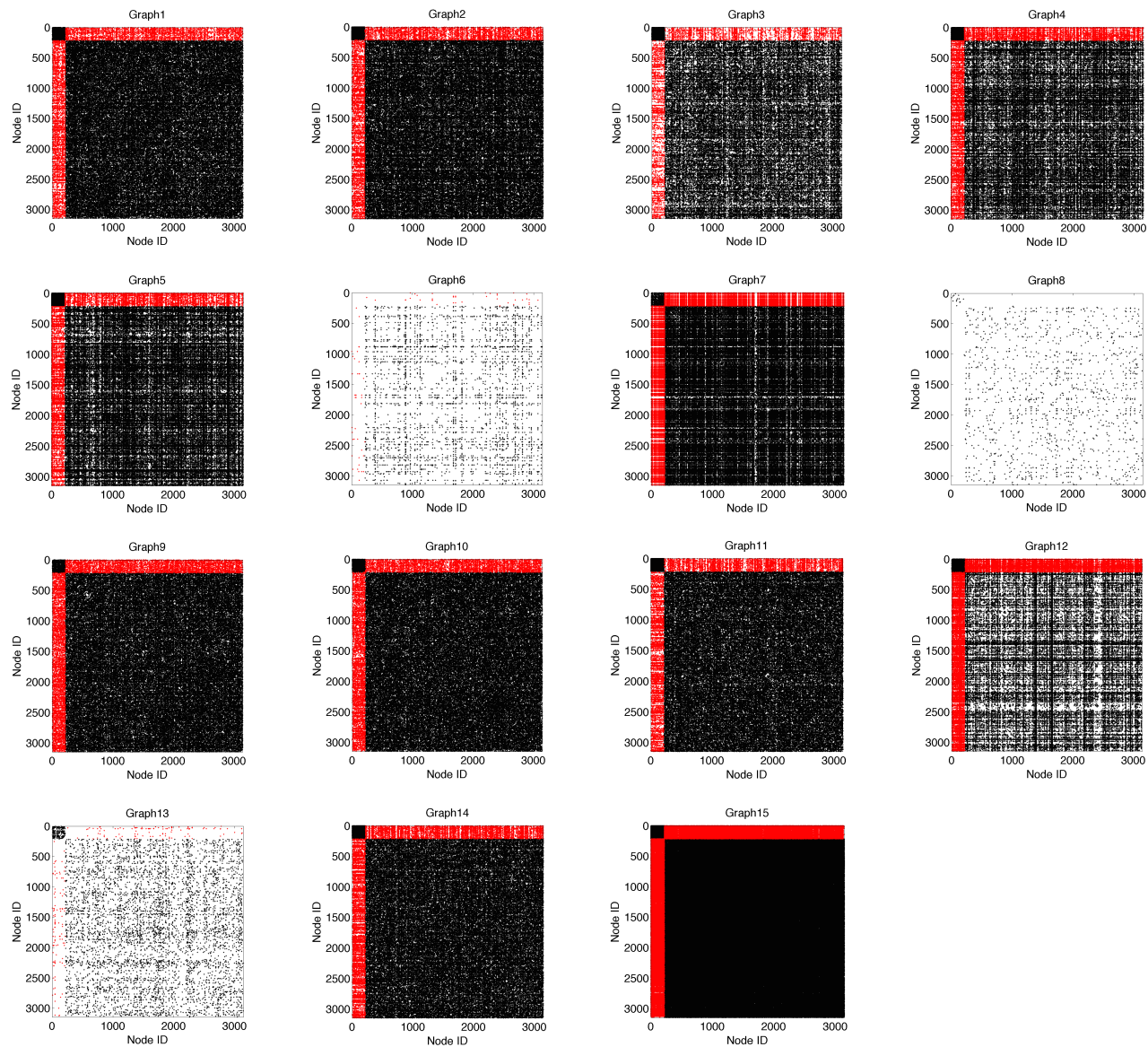
A.2 Protein *Protein* [17] consists of 5 relational graphs of yeast proteins. These relations are co-participation in a protein complex, Pfam domain structure similarity, protein-protein interactions, genetic interactions and gene profile similarity. Proteins that exhibit *transport facilitation* function are labeled as positives. The remaining proteins are labeled as negative.



A.3 Gene1 *Gene1* [12] contains 15 relational graphs among yeast genes. All the genes are labeled according to *Gene Ontology (GO)* association file from the *Saccharomyces Genome Database*. For binary classification setting, we choose the label with maximum number of genes in *Cellular Component (CC)* domain as positive class and consider the remaining genes as negative class.



A.4 Gene2 We construct *Gene2* in a similar way as for *Gene1*. The major difference is that the labels in this dataset come from *Molecular Function (MF)* domain.



B Precision-Recall Plots on Real-World Multi-graphs

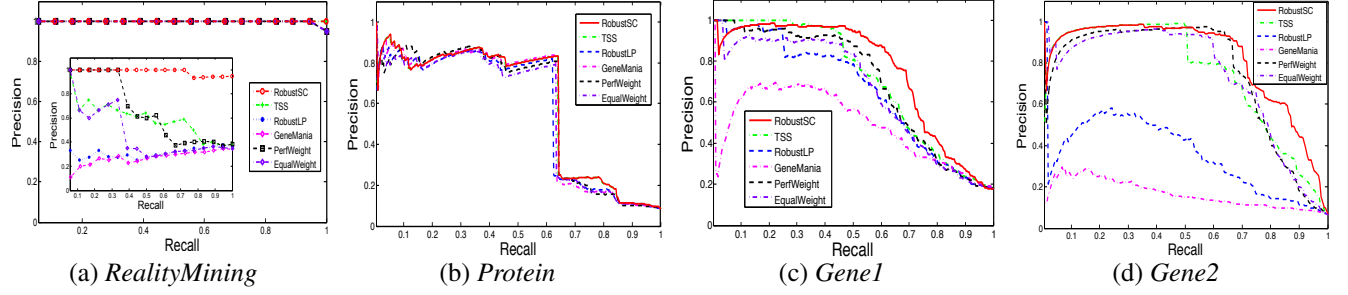


Table 3: Precision vs. Recall of competing methods in four real-world multi-graphs: (a) *RealityMining* (4 graphs), (b) *Protein* (5 graphs), (c) *Gene1* (15 graphs), and (d) *Gene2* (15 graphs). Inset plot in (a) shows performance when 6 rewired graphs with low intensity are injected to *RealityMining* multi-graph—noise hinders existing methods dramatically, whereas ROBUSTMULTISC remains near-stable.

C Additional Noise-Test Results

We provide additional noise-testing results on *RealityMining* with 2 and 4 injected graphs, under 3 noise models and 2 intensity levels in Table 4.

Table 4: Performance of all methods on real-world multi-graphs. RM also injected with 2 and 4 intrusive graphs with various settings. Values depict mean and standard deviation Average Precision (AP) (over 10 runs with different labeled set).

Dataset	#Graphs	NoiseModel	Intensity	PROPOSED	PERF-WGHT	EQL-WGHT	TSS	ROBUSTLP	GENEMANIA
RM	4+2	Adversarial	Low	0.970±0.004	0.707±0.050	0.554±0.049	0.525±0.075	0.851±0.025	0.470±0.097
	4+2	Adversarial	High	0.970±0.004	0.611±0.049	0.484±0.045	0.554±0.048	0.809±0.044	0.359±0.068
	4+2	Rewire	Low	0.970±0.004	0.695±0.054	0.669±0.050	0.718±0.061	0.873±0.021	0.509±0.086
	4+2	Rewire	High	0.970±0.004	0.563±0.064	0.537±0.062	0.646±0.066	0.824±0.045	0.290±0.050
	4+2	Erdos-Renyi	Low	0.970±0.004	0.905±0.024	0.841±0.042	0.657±0.081	0.928±0.011	0.773±0.067
	4+2	Erdos-Renyi	High	0.970±0.004	0.942±0.016	0.920±0.022	0.895±0.042	0.930±0.015	0.883±0.055
	4+4	Adversarial	Low	0.970±0.004	0.531±0.042	0.372±0.040	0.390±0.047	0.427±0.063	0.260±0.029
	4+4	Adversarial	High	0.970±0.004	0.383±0.045	0.297±0.025	0.576±0.057	0.339±0.051	0.215±0.010
	4+4	Rewire	Low	0.930±0.004	0.610±0.062	0.503±0.058	0.561±0.062	0.505±0.066	0.319±0.046
	4+4	Rewire	High	0.907±0.004	0.437±0.054	0.349±0.044	0.542±0.062	0.334±0.048	0.217±0.011
	4+4	Erdos-Renyi	Low	0.970±0.004	0.867±0.029	0.770±0.045	0.482±0.084	0.869±0.034	0.698±0.094
	4+4	Erdos-Renyi	High	0.970±0.004	0.942±0.016	0.917±0.024	0.659±0.078	0.925±0.026	0.834±0.068

D Graph Removal Order of ROBUSTMULTISC on Real Datasets

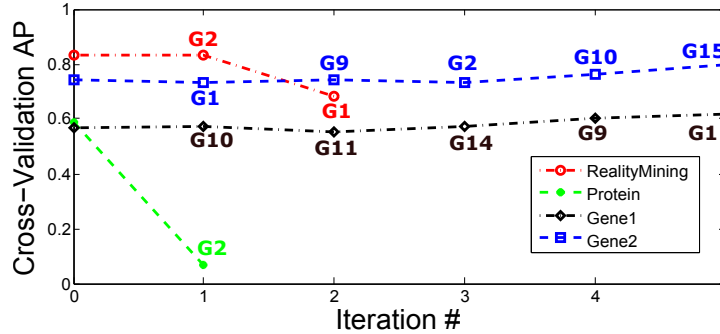


Figure 7: The removal order of individual graphs from real-world multi-graphs during the course of ROBUSTMULTISC, which removes G_x from *RealityMining* (See A.1) and G_y from *Protein* (See A.2). It also removes 5 (uninformative) graphs each from both *Gene1* (A.3) and *Gene2* (A.4), where the cross-validation performance increases slightly.

E Inferred Weights on All Datasets

In the following we provide the inferred weights by all the six methods on all the ten datasets as listed in Table 2.

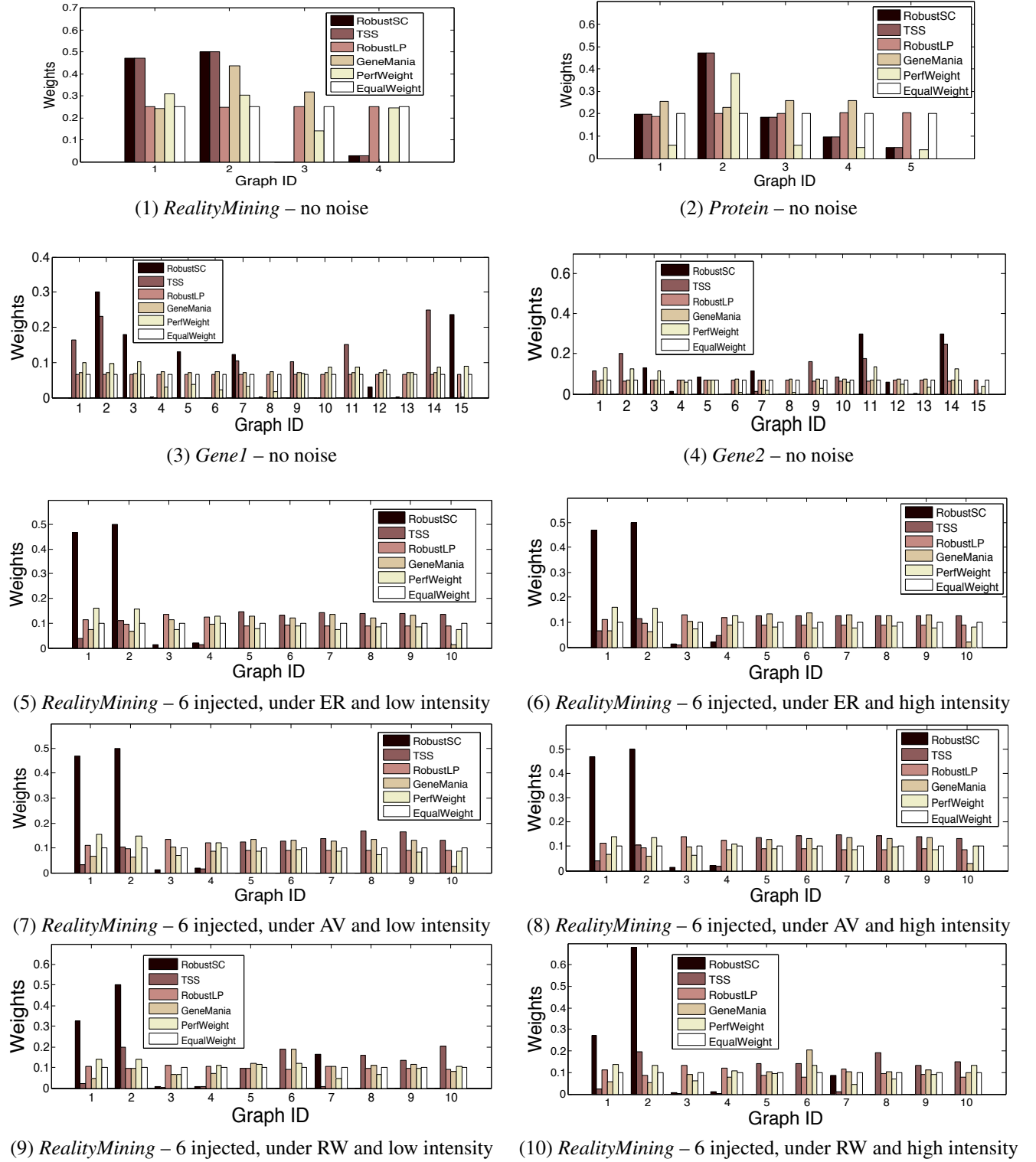


Figure 8: Inferred graph weights (normalized) on all ten datasets by all six methods.